



Load Balancing via Parallel Hypergraph Partitioners

Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson
Sandia National Laboratories, Albuquerque
kddevin@sandia.gov

Unit Çatalyürek
Ohio State University, Columbus

Rob Bisseling
Utrecht University, The Netherlands

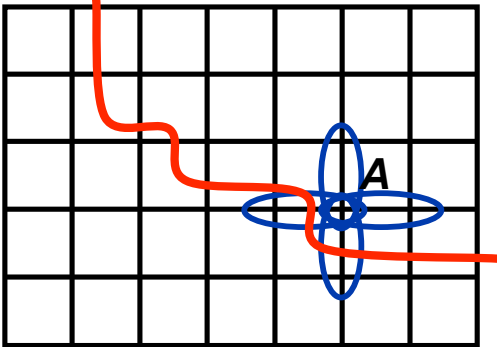
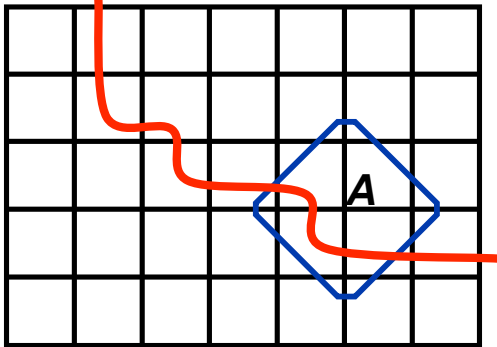


Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.





Graph Partitioning vs. Hypergraph Partitioning

Graph Partitioning	Hypergraph Partitioning
Kernighan, Lin, Schweikert, Fiduccia, Mattheyses, Simon, Hendrickson, Leland, Kumar, Karypis, et al.	Alpert, Kahng, Hauck, Borriello, Çatalyürek, Aykanat, Karypis, et al.
Vertices: computation.	Vertices: computation.
Edges: two vertices.	Hyperedges: two or more vertices.
Edge cuts approximate communication volume.	Hyperedge cuts accurately measure communication volume.
Assign equal vertex weight while minimizing edge cut weight.	Assign equal vertex weight while minimizing hyperedge cut weight.
	



Impact of Hypergraph Partitioning

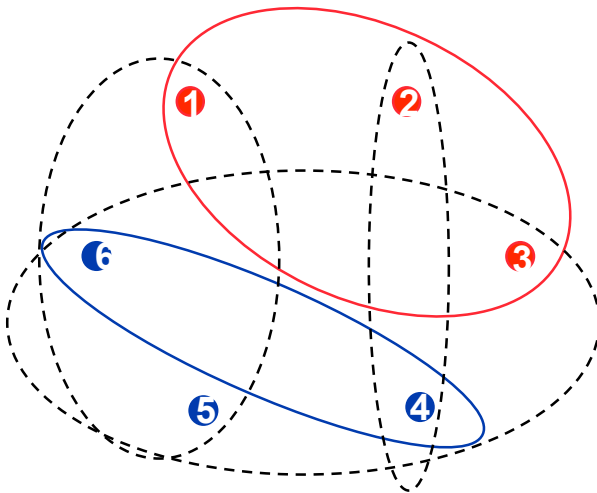
- **Greater expressiveness \Rightarrow Greater applicability.**
 - Structurally non-symmetric systems
 - circuits, biology
 - Rectangular systems
 - linear programming, least-squares methods
 - Non-homogeneous, highly connected topologies
 - circuits, nanotechnology, homeland security databases
- **Accurate communication model \Rightarrow lower application communication costs.**
- **Several serial hypergraph partitioners available.**
 - hMETIS (Karypis)
 - PaToH (Çatalyürek)
 - Mondriaan (Bisseling)
- **Parallel partitioners needed for large, dynamic problems.**
 - Zoltan PHG (Sandia)
 - Parkway (Trifunovic)



Matrix Representation

- View hypergraph as matrix (Çatalyürek & Aykanat)
 - Vertices == columns
 - Edges == rows
- Communication volume associated with edge e :

$$CV_e = (\# \text{ processors in edge } e) - 1$$
- Total communication volume = $\sum_e CV_e$

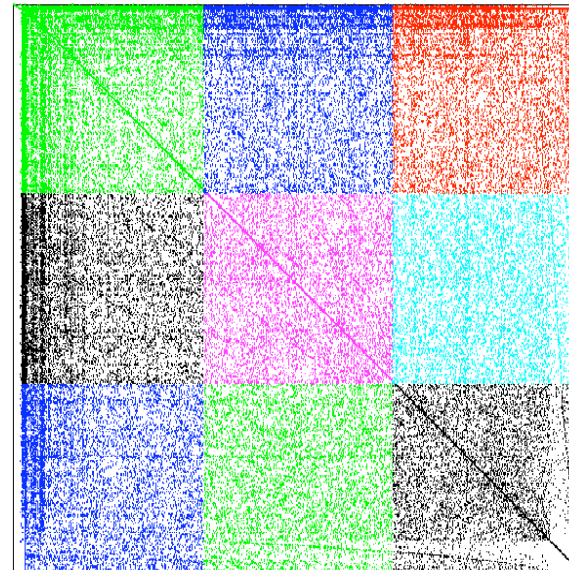


$$\begin{pmatrix} y \\ y \\ y \\ y \\ y \end{pmatrix} = \begin{pmatrix} * & * & * & & \\ & * & & * & \\ * & & & * & * \\ & & * & * & * & * \\ & * & * & * & * & * \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}$$



Data Layout

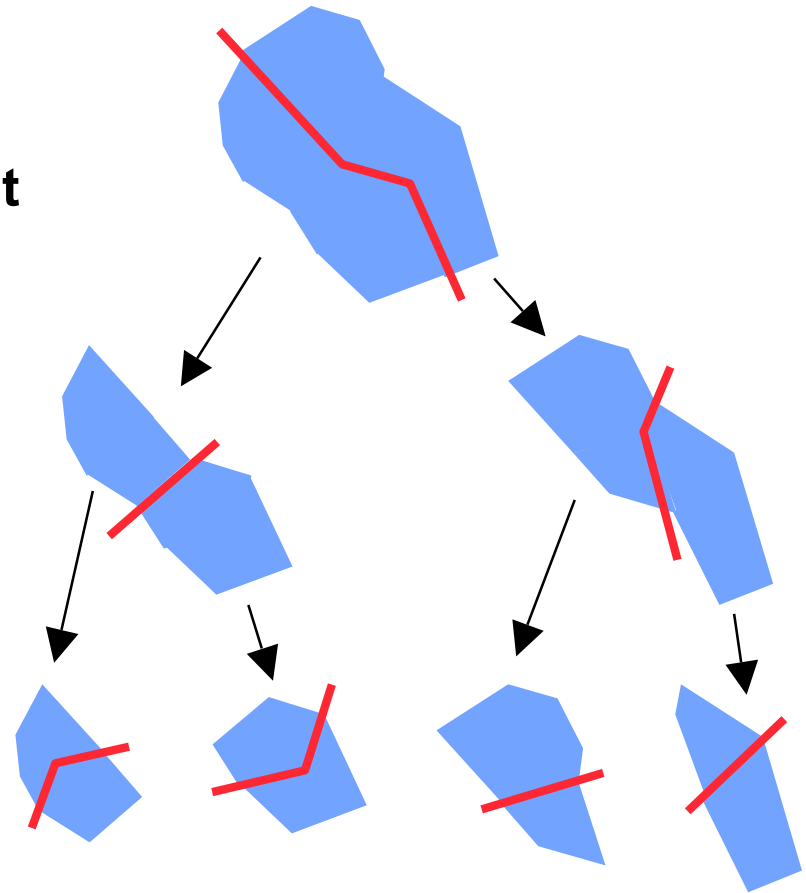
- **2D data layout** within hypergraph partitioner.
 - Does not affect the layout returned to the application.
 - Vertex/hyperedge broadcasts to only \sqrt{P} processors.
 - Maintain scalable memory usage.
 - No “ghosting” of off-processor neighbor info.
 - Differs from parallel graph partitioners and Parkway.
 - Design allows comparison of 1D and 2D distributions.





Recursive Bisection

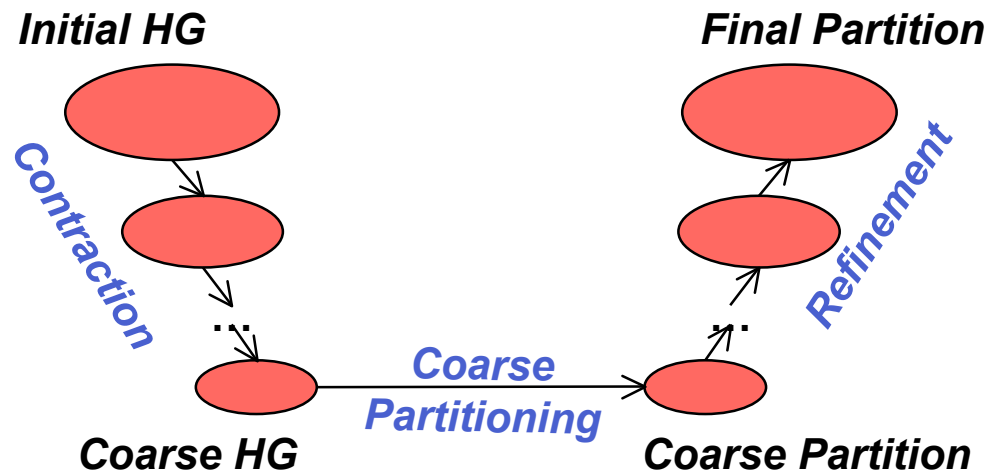
- **Recursive bisection approach:**
 - Partition data into two sets.
 - Recursively subdivide each set into two sets.
 - Only minor modifications needed to allow $P \neq 2^n$.
- **Two implementation options:**
 - Split *only the data* into two sets; use all processors to compute each branch.
 - Split *both the data and processors* into two sets; solve branches in parallel.





Multilevel Scheme

- Multilevel hypergraph partitioning (Çatalyürek, Karypis)
 - Analogous to multilevel graph partitioning (Bui&Jones, Hendrickson&Leland, Karypis&Kumar).
 - **Contraction**: reduce HG to smaller representative HG.
 - **Coarse partitioning**: assign coarse vertices to partitions.
 - **Refinement**: improve balance and cuts at each level.



Multilevel Partitioning V-cycle



Contraction

- Greedy maximal weight matching algorithms.
- Heavy connectivity matching (Çatalyürek)

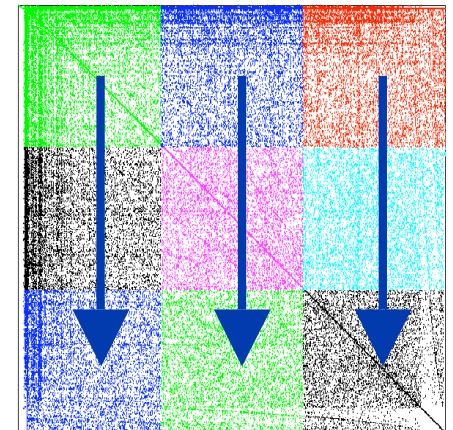
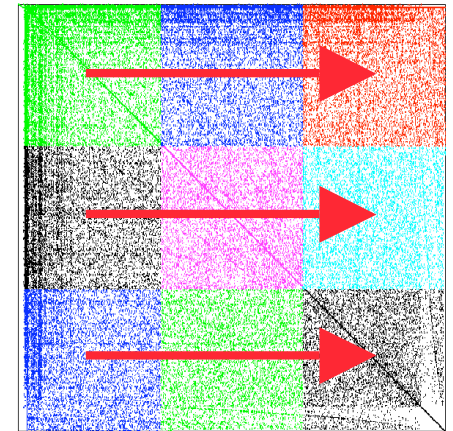
Inner-product matching (Bisseling)

- Match columns (vertices) with greatest inner product \Rightarrow greatest similarity in connectivity.



Parallel Matching in 2D Data Layout

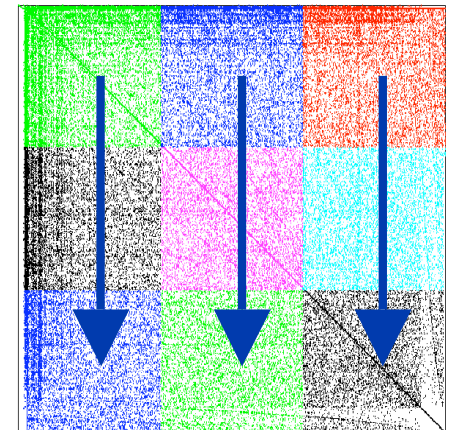
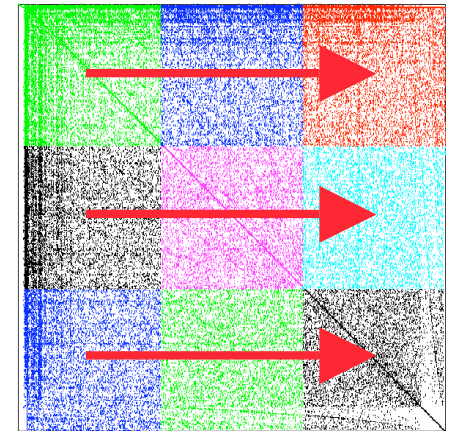
- On each processor:
 - Broadcast subset of vertices (“candidates”) along processor row.
 - Compute (partial) inner products of received candidates with local vertices.
 - Accrue inner products in processor column.
 - Identify best local matches for received candidates.
 - Send best matches to candidates’ owners.
 - Select best global match for each owned candidate.
 - Send “match accepted” messages to processors owning matched vertices.
- Repeat until all unmatched vertices have been sent as candidates.





Coarse Partitioning

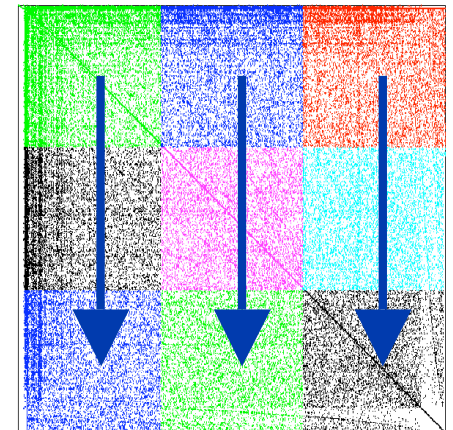
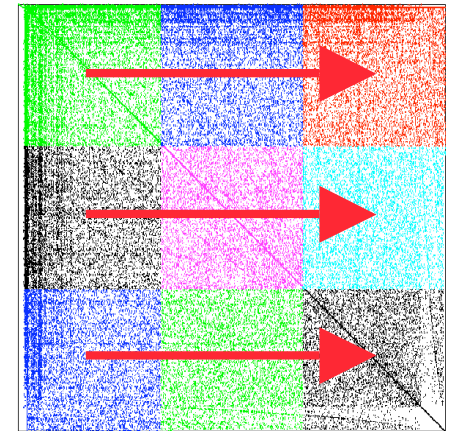
- Gather coarsest hypergraph to each processor.
 - Gather edges to each processor in column.
 - Gather vertices to each processor in row.
- Compute several different coarse partitions on each processor.
- Select best local partition.
- Compute best over all processors.





Refinement

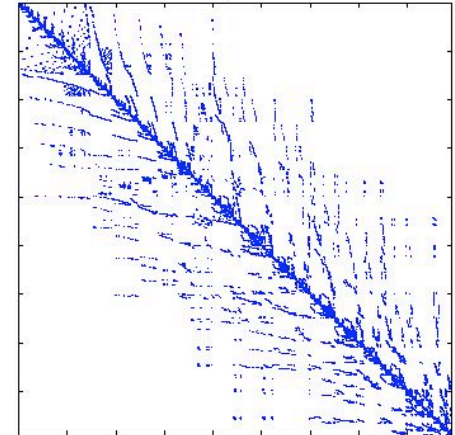
- For each level in V-cycle:
 - Project coarse partition to finer hypergraph.
 - Use local optimization (KL/FM) to improve balance and reduce cuts.
 - Compute “root” processor in each processor column: processor with most nonzeros.
 - Root processor computes moves for vertices in processor column.
 - All column processors provide cut information; receive move information.





Graph vs. Hypergraph Partitioning

- **Cage12: Cage model of DNA electrophoresis (van Heukelum in U. FL. Matrix Collection)**
 - 130,228 rows & cols; 2,032,536 nonzeros.
 - 64 partitions.
- Hypergraph partitioning reduced communication volume by 8-17%.
- Zoltan PHG comparable to PaToH.



Partitioning Method	One Processor		64 Processors	
	Comm. Volume	Time (secs.)	Comm. Volume	Time (secs.)
Graph (METIS/ParMETIS)	198,362	1.5	222,316	2.2
Serial HG (PaToH)	182,345	32.7	NA	NA
Parallel HG (Zoltan PHG)	183,027	32.0	184,861	9.3



Zoltan-PHG

Performance Results

- As number of processors increases:
 - Communication volume (CV) does not degrade.
 - Execution time is reduced (but speedup not yet perfect).

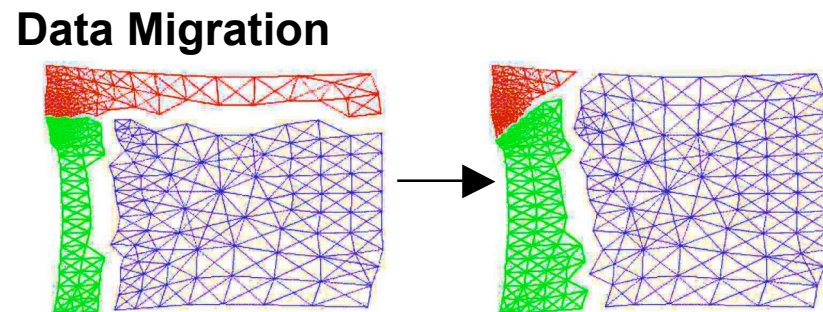
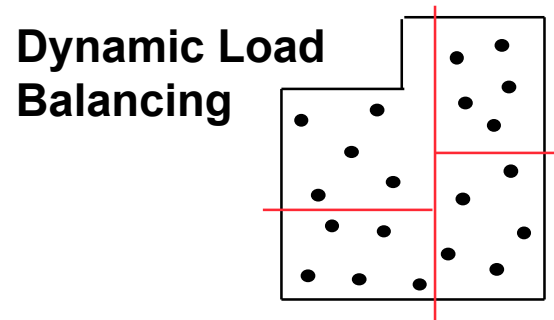
64 partitions on:	<i>P</i> = 1		<i>P</i> = 4		<i>P</i> = 16		<i>P</i> = 64	
Matrix (size; # nz)	CV	Time	CV	Time	CV	Time	CV	Time
PolymerDFT (46K x 46K; 2.7M)	86,204	33.9	86,072	11.8	85,726	5.0	85,863	4.3
IBM18 circuit (202K x 211K; 820K)	29,991	19.7	30,037	9.5	30,306	5.9	30,722	7.2
Random (1M x 1M; 20M)	---	---	25.3M	6344	25.3M	1572	25.4M	731
Voting175 Markov (1.1M x 1.1M; 6.7M)	147.3K	137	148.1K	64	148.0K	40	149.0K	30
Cage14 (1.5M x 1.5M; 27M)	1.60M	832	1.61M	481	1.61M	264	1.62M	161



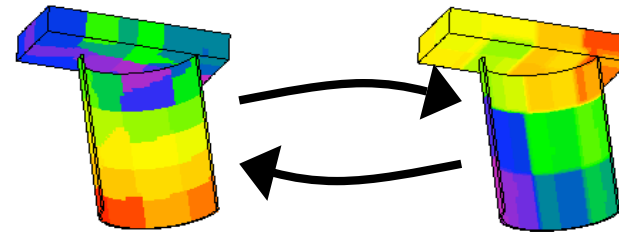
The Zoltan Toolkit

Data services for unstructured, dynamic and/or adaptive computations.

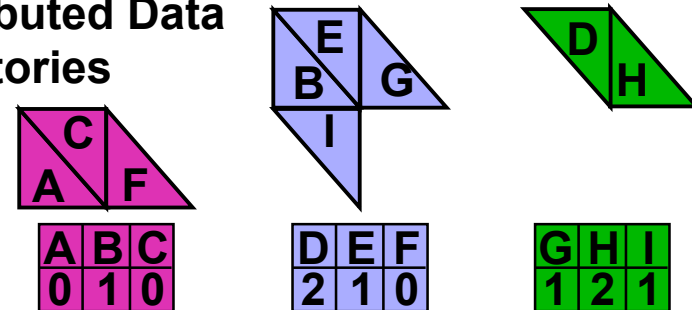
<http://www.cs.sandia.gov/Zoltan>



Unstructured Communication



Distributed Data Directories



Dynamic Memory Debugging





Future Work

- **Increase speed while maintaining quality.**
 - Heuristics for more local, less expensive matching
 - Parallel coarse partitioning
 - K-way refinement
- **More evaluation of current design.**
 - 1D vs. 2D data layouts
 - During recursive bisection, split only data or both processors and data
- **Incremental partitioning for dynamic applications.**
 - Minimize data migration.
- ***Watch for release in Zoltan later this year!***